The Pragmatic Evaluation of Tool System Interoperability

Aldo de Moor

CommunitySense
Cavaleriestraat 2, 5017 ET Tilburg, the Netherlands
ademoor@communitysense.nl**

Abstract. Collaborative communities are increasingly supported by systems of information and communication tools. Much current research and development focuses on the technical and semantic aspects of tool system interoperability. However, for developing effective tool systems, their pragmatic evaluation is also essential. This implies that the usage context of tools is taken into account. In this paper, we envision an approach to the pragmatic evaluation of tool system interoperability. Its main elements are a conceptualization of the tool system, its usage context, and an evaluation processs. With these basic elements, pragmatic evaluation can be operationalized in many different ways. We illustrate our approach with several real-world examples.

1 Introduction

Information systems development in the classic sense of waterfall-based requirements analysis, design, and implementation of monolithic and customized information systems more or less from scratch is on the way out. Collaborative communities, in which people work together to accomplish common goals, increasingly make use of a wide range of (Internet-based) information and communication tools to support their information processing, communication, and coordination needs. A typical configuration for such a community includes a content management system, a couple of mailing lists, discussion fora, chat or conferencing software, co-authoring tools, a calendar tool, and so on. However, this is only a minimal set of tools. Most communities use a wide range of additional tools, very much depending on the particular domain they are in[1].

We define a *tool system* as the set of integrated and customized information and communication tools tailored to the specific information, communication, and coordination requirements of a collaborative community. There are numerous, partially overlapping implementations of such functionalities. In addition, many of these tools are built on top of an emerging cyberinfrastructure of organizational practices, technical infrastructure, and social norms [14]. Furthermore, each community has its own, unique

** Published in Pfeiffer, H.D., Kabbaj, A. and Benn, D. (eds.), Proc. of the 2nd ICCS Conceptual Structures Tool Interoperability Workshop (CS-TIW 2007), Sheffield, UK, July 22, 2007 (invited paper). Research Press International, Bristol, UK, pp.1-19

[1] A very good example are disaster response communities. One domain-specific tool that has proven its worth is the Sahana open source disaster management system, which was developed in Sri Lanka after the 2004 tsunami: http://www.sahana.lk

way of using these functionalities. Finally, the requirements and technologies used are in constant flux. In all, this makes it extremely complicated to come up with standardized prescriptions for the best tool system for a particular community at a particular moment in time. New forms of analysis, roles in software development, and the meaning of use and maintenance need to evolve [28]. To design useful information systems by selecting the right components, available functionalities in the form of modules and services need to be evaluated in their context of use by the communities of use themselves.

A tool system is all about synergy and alignment. How to create a well-tuned orchestra out of these tools, which is able to perform a magnificent symphony? Like in a human orchestra, this should go much beyond listing a set of the technical abilities of the orchestra members. Each member has her own strengths and subtleties, which cannot be exactly described, but needs to be experienced. Then, after much practice and trial-and-error, the ensemble of all these unique individuals starts to express yet another layer of synergetic qualities, distinct for different pieces played. Everybody would agree that to assess the strengths and weaknesses of the various orchestra members, to know which pieces they are best able to play, and to create the right combination of performers, the trained ear of an experienced conductor is indispensable. Yet, in the design of tailored tool systems, arguably similarly complex, the conductor is missing.

In systems development for collaborative communities, the real difficulty is not in describing the functionalities of the individual tools, but in assessing the *interoperability* of a configuration of tools in a particular usage context. Much has been said about interoperability. The most general definition is that interoperability is the need to make heterogeneous information systems work in the networked world [33]. One that is better suited for our purpose of the assessment of tool system interoperability is the one given by Miller in [24], who defines it as the ongoing process of ensuring that the systems, procedures, and culture of an organisation are managed in such a way as to maximise opportunities for exchange and re-use of information, whether internally or externally. In such a view of interoperability, finding a way to represent and analyze the *usage context* is all-important.

Much interoperability research stems from the domain of component-based systems development and web services. The importance of syntactic and semantic interoperability has been recognized for quite a while already [29, 17]. For example, the Universal Description, Discovery and Integration (UDDI) standard provides rules for building service directories and the facilitation of top-down querying capabilities [34]. However, although syntactic and semantic interoperability is necessary, also the context-dependent (i.e. *pragmatic*) needs of the users should be taken into account by more explicitly linking service description, discovery, and invocation to context [30].

The key questions we will address in this paper are the following: how to conceptualize the idea of usage context in tool system interoperability evaluation? What would an evaluation procedure look like? What would the results of such a procedure be in terms of design choices? The goal of the paper is to come up with a ”minimal conceptual model”, an ontology as it were of the main concepts and relations needed to develop pragmatic evaluation methods. Such a model can be used in the construction of actual methods tailored to the specific needs of real-world user communities. It can

be used as a framework for the construction and comparison of such methods, ensuring that they are truly pragmatic in the semiotic sense of the word.

In this paper, we will synthesize and extend a body of recent work. In [8], we examined the role of pragmatic patterns in the evolution of semantic resources. In [7], we presented a basic conceptual model of tool system interoperability. We proposed a practical community-driven courseware evaluation approach in [10]. Together, these building blocks are the foundation of the pragmatic evaluation approach proposed in this paper.

In Sect. 2, we give an informal introduction into the kind of issues at play in pragmatic tool system interoperability by exploring a real-world example of a tool system used to support the co-authoring of a call for papers. In Sect. 3, we conceptualize the tool system. Sect. 4 captures the usage context. In Sect. 5, we model the evaluation process. We end the paper with a discussion and conclusions in Sect. 6 and 7.

2 Pragmatic Tool System Interoperability Evaluation in Practice: Co-Authoring a Call for Papers

Community interaction processes tend to be very complex, leading to context-specific requirements which often cannot be met completely by any particular tool. Vice versa, many tools are often used for a very different purpose than what they were originally designed for. Thus, the tool systems emerging to support collaboration are about the process of articulating ways how to *use* the tools as much as about the functionalities themselves, if not more so. Next, we will describe a powerful real-world example of how relatively straightforward tools can be combined into tool systems that as a whole provide very complex but useful functionality[2].

In 2006, the author was one of the three co-chairs of the First International Pragmatic Web Conference. Since the chairs live and work far apart, they tried to write the Call for Papers by sending Word-files around. However, as the Pragmatic Web is such a new paradigm, confusion abounded and versions kept flying back and forth between the three of them. It was clear that just using e-mail and word processors did not fit their collaborative needs.

The co-chairs, additionally, had a need for voice contact. However, the phone would not do, since the three of them needed to discuss simultaneously. Phone conferences were still relatively expensive, so they decided to use Skype instead. This voice-over-Internet application at the time allowed up to five people to discuss for free. Additional benefits were that the sound quality is much better than that of the phone, and that one can talk using a microphone. This means that one's hands are free and one can type while talking.

Although the co-chairs could now talk simultaneously, they were not there yet. They were considering three versions of the document: one previously written by one chair, one by another chair, and the third one the version-in-progress which contained the modifications that they were making as they spoke. This was too much, as all of them were getting very confused by all those almost-but-not-quite similar texts. Instead, they came up with the following solution (Fig. 1).

[2] Previously described on http://growingpains.blogs.com/home/2006/03/it_takes_at_lea.html

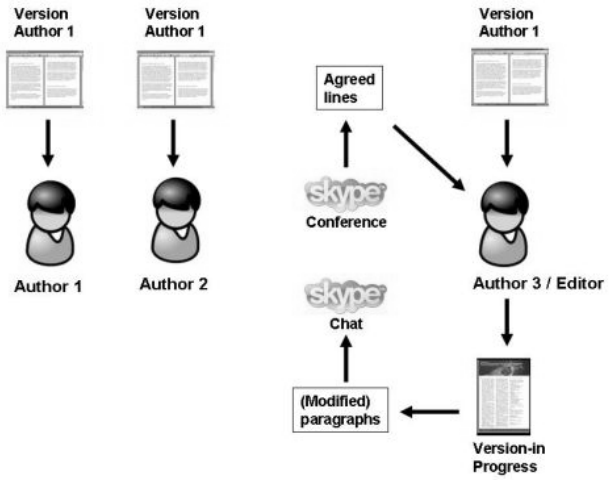

Fig. 1. A Tool System for Co-Authoring a Call for Papers

- All co-chairs opened one of the previous versions in Word for reference.
- One co-chair acted as editor, and opened the version of the second chair as the starting point of the version-in-progress.
- The co-chairs started editing this version paragraph-by-paragraph, line-by-line. Each time, the editor would copy the paragraph currently being discussed, and paste it into Skype's chat-window.
- All three co-chairs would look at this paragraph, and compare it with the same paragraph in the other Word-version. They could then propose and discuss modifications a few lines at a time. The editor kept track of these changes in the version being edited. Having agreed orally upon only a few lines, it was already getting difficult to see the bigger picture. Whenever that was the case, the editor copy/pasted the now modified paragraph into Skype-chat again. The co-chairs would then - orally - make a few more changes, to be copy/pasted into Chat by the editor.
- This continued until all co-chairs were satisfied with the full paragraph. They then moved on to the next paragraph, until they were, at last, happy with the final Call for Papers.

So, what exactly happened? A few standard *tool functionality components* were used: Word for editing text; Skype-Conference for group voice discussion; Skype-Chat for keeping track of the rapidly changing "focus-text". Two *roles* (Author, Editor) were distinguished. A few simple *process rules* were adhered to: all authors focus on the same paragraph; all authors compare the current focus-paragraph with the related Word-paragraph and make comments; the editor changes lines on which consensus has been reached in the version-in-progress; the editor pastes modified paragraphs into chat whenever there is too much confusion. This arrangement then allowed for a

very complex group authoring process to be successfully supported by information and commnunication technologies. For all the co-chairs, it has been a transformative experience to see how such a dauntingly complex collaborative task could be done online instead of face-to-face. This experience reconfirms the idea that the future of collaboration support will not be in very expensive, proprietary collaborative platforms. Instead, continuously rediscovering the right combination of tools, process, and context seems to be the direction to go.

3 A Conceptual Model of the Tool System

A functionality is a set of functions and their specified properties that satisfy stated or implied needs[3]. Functionalities come in different forms of aggregation, from simple word processing functions to complete modules and applications supporting particular workflows or business processes. When evaluating functionality, the right level of granularity should be chosen.

In [7], we introduced a basic model of a *tool system*, consisting of components of different levels of granularity. At the lowest level of granularity, we distinguish *systems* of tools or services. The next level consists of the *tools* or services themselves. Then come the *modules* comprising the tools or services. Finally, we distinguish the particular *functions* grouped in a module.

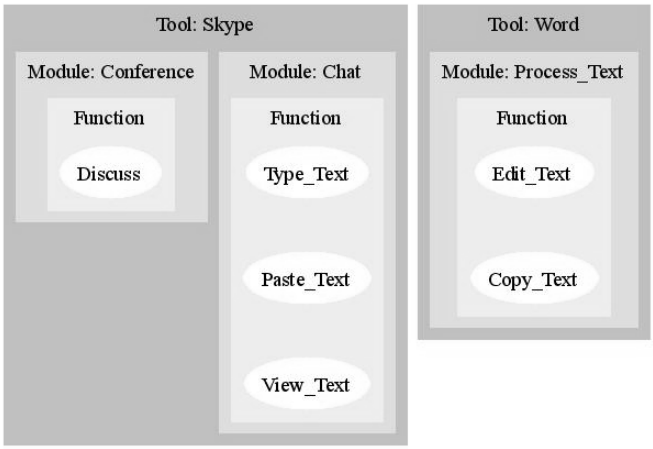

Fig. 2. A Functionality Decomposition of the Co-Authoring Tool System

[3] Definition by the Software Engineering Insitute Open Systems Glossary: http://www.sei.cmu.edu/opensystems/glossary.html

Functionality components of different levels of granularity can be linked in many ways. The *interface* between two functionality components A and B is described by the externally visible representation of the knowledge structures used in A and B.

In [7], we focused on the communication aspects of the interoperating functionality components. We abstracted from the internal *information processes*, which are operations on information objects within a functionality component, such as the creation, modification, or deletion of an object. For tool selection, such processes are important, however, so we include them in our conceptual model. Note that we are only interested in those information processes visible to a user, so we abstract from all internal computational processes.

Fig. 2 gives a functionality decomposition of the main elements of the co-authoring support system described in the previous section. For simplicity, information objects and processes have not been included. The Skype Conference-module includes the function that allows a group of users to discuss. Its Chat-module allows users to type text, paste text from the clipboard onto the chat screen, and to view the typed and pasted text. One of the main modules of Word is to support the processing of text, including editing a text file and copying text onto the clipboard.

Strictly speaking, a tool system conceptualization is only about functionality, not about context. However, the tools, modules, and functions selected are a subset of all potentially available functionality. This selection is determined by the usage context. This is a good example of how the Tool System and the Human System are co-dependent and co-evolve, to use Doug Engelbart's terms [15].

This leads us to the question how to conceptualize the usage context? It is clear that a sense of communal purpose is important, but how exactly to operationalize this notion and its linkages to the tool system?

4 A Conceptual Model of the Usage Context

In [8], we made a first stab at modelling the pragmatic layer of web-based information systems. We claimed that there needs to be a user-controlled selection process of semantic representations[4]. We defined a pragmatic context to consist of a common context and a set of individual contexts. The common context was defined by the common concepts and conceptual definitions of interest to a community, the communicative interactions in which these concepts are defined and used, and a set of common context parameters (relevant properties of concepts describing the context). Each community member was said to have an *individual context*, consisting of individual concepts and definitions of interest, as well as individual context parameters. We then characterized the pragmatic context by a set of pragmatic patterns, including individual and common pragmatic patterns. In that paper, we focused on the meaning negotiation process between community members and did not further explore the notion of the common and individual context parameters, nor their conceptualization in patterns. In the current paper, the focus is

[4] Although in the current paper, the focus is on functionality selection rather than semantics resources selection, this distinction is not relevant here. In both cases, we are interested in the ontological status of resources/tools, and how their relevance to communities of users can be assessed.

reversed: we do not so much look at how individual and common pragmatic patterns influence the meaning negotiation process. Rather, we further explore the make-up of the individual and common pragmatic patterns, and how they come about in the first place. In that sense, the papers complement each other.

4.1 Goals

Goals or objectives are crucial in the pragmatic view. Everything starts with goals. Goals give a sense of purpose, drive people and processes, and can be used as evaluation criteria.

We distinguish two types of goals. First, there are *activities*, such as "writing a call" or "making a group assigment". Such activities in fact are operationalized goals: processes with a concrete deliverable as an outcome, often in the form of an information object such as a report or a message. However, many goals are abstract and cut right across processes and structures. Examples of such goals are non-functional requirements and quality criteria, like "security", "interactivity", and so on. We call such abstract goals *aspects*.

Although activities can be viewed as workflows, we abstract from their design and implementation details, such as concurrency and sequence. Although definitely important in the final construction of the information system, for the purpose of functionality component selection they add unnecessary complexity. This initial stage, the focus of this paper, concerns itself much more with selection of *potential* functionalities, not their actual configurations for workflow support.

4.2 Actors

Many stakeholders are involved in tool system evaluation. For example, in the domain of courseware evaluation, students, teachers, the computer center, etc. all have their very specific, often partially incompatible interests, needs, and goals. Students prefer easy-to-use functionalities, whereas main concerns of the computer center include security and reusability [10]. In order to ensure that all requirements are captured, contrasted, and balanced, it is not sufficient to examine "The User". Instead, a very careful inventory needs to be made of the actor roles that the various stakeholders play. Making roles explicit is gaining prominence in, for instance, the Role Based Access Control paradigm[5] as a way to systematically determine actor responsibilities in workflows and access to functionalities and information resources. Most role classifications are quite abstract and technology-focused (Administrator, Facilitator, Member etc.). However, many other typologies are possible, often quite specific to a particular domain. Roles could, for instance, be based on workflow process (Author, Editor, Reviewer, ...), organizational structure (Secretary, Manager, ...), or based on the main stakeholders in a particular domain (UNEP, Corporation, NGO, ...). Carefully defining the responsibilities, permissions, and prohibitions attached to these roles helps in better defininig the use patterns of information systems. Besides operational status, such actor roles should also be explicitly involved in the development of the socio-technical systems-in-use, as is the focus of this paper

[5] http://csrc.nist.gov/rbac/

4.3 Domain

A third layer of the model is the domain in which the collaborative community using the tool system (inter)operates. The domain is an important, but still ill-understood element in tool system evaluation. Aspects that influence evaluation processes and functionalities of the tool system include issues of structure and size, for example is it a distributed network or centralized organization, a large or a small organization? What setting is the tool being used in: an academic, a corporate, a governmental, or a non-governmental setting? What is the financial situation: are there resources for acquisition or customization of software, or is off-the-shelf, open source software the only option? Are there political alliances and commitments that force or preclude the use of certain software? In other words, the domain determines how many degrees of freedom there are in evaluation process.

5 The Tool System Interoperability Evaluation Process

Summing up, we have established tool functionality components, goals (activities and criteria), actor roles played by users, and the domain as important primitives playing a role in the evaluation process of tool system interoperability (Fig. 3). We now examine the structure of this process itself.

The stakeholders playing actor roles are usually very busy, and have little time and opportunity to develop an in-depth knowledge of the technology at hand nor to make many requirements explicit. Each human stakeholder possesses a treasure trove of subtle tacit knowledge of the tool system-in-use. Tacit knowledge is highly personal and is deeply rooted in action, procedures, routines, commitment, ideals, values and emotions, and is thus hard to formalize [20]. Creating massive, unwieldy evaluation processes forcing users to assess numerous issues in detail is a dead-end road. Rather, a useful method should focus on a few, comprehensive goals, which allow for commonalities and differences in evaluations to be easily established, while giving ample opportunity for efficiently zooming in on the underlying factors in human discussion processes. The core goals could thus be used as a sieve, allowing expensive human interactions to be focused immediately on the most interesting issues of shared understanding and disagreement.

Although the number of evaluation goals to be made explicit should be minimized, a well-structured evaluation process is needed, respecting the different perspectives of the stakeholders. The reasons for making final selection decisions need to be clearly conveyed to the users to increase legitimacy and technology acceptance. Decisions on functionality component selection and configuration should be devolved to the lowest actor-level possible. For example, in a courseware setting, overall class functionality choices will generally need to be made by the lecturer, group settings by the groups, and individual settings by the individual user.

5.1 Evaluation Subprocesses

To structure the evaluation process, we subdivide it into three subprocesses: use, interpretation, and decision making.

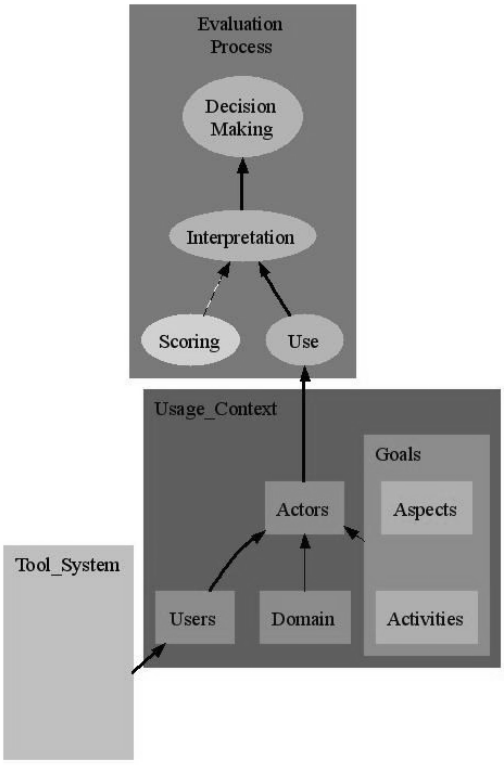

Fig. 3. The Pragmatic Tool System Interoperability Evaluation Process

- In the *use process*, users get hands-on experience with the functionalities. Evaluations can be either ex-ante or ex-post this use process. Ex-ante evaluations may involve scenarios: task-based expressions of human-machine interaction, which help users to at least partially imagine the potential usefulness and acceptability of functionalities [32]. Ex-post evaluations concern evaluations done after the functionality has been actually used. Although ex-ante evaluations can be useful, ex-post evaluations are preferred, since unexpected affordances and constraints of functionalities often only become apparent during actual use [37].
- The *interpretation process* is the process in which the users score and then use the scores to prioritize and discuss the usefulness of the tool system (as there are many ways to obtain scores, and as the scoring process is the most formal and quantitative stage of the interpretation process, it will be further discussed in Sect. 5.2). It is crucial that the interpretation process is both effective and efficient. The interpretation process needs to be *effective*, in that - ideally - the interests and goals of all relevant stakeholders are satisfied. This requires that the right stakeholders are involved in the evaluation, and that they are provided with an instrument to actively identify, compare, and discuss their requirements and interests. Carefully designed strategies

are needed to balance the often competing demands of many different stakeholders [2]. The evaluation process should also be *efficient*. Empirical findings have shown that users have more confidence in simple information and communication technology (ICT) evaluation methods [5]. If evaluations become more numerous, and are to be performed by very busy professionals who often have other priorities, only the most relevant questions should be asked. This could be done in the form of an initial quick-scan, with more detailed analysis of areas identified to be problematic only to be done in a follow-up analysis. *Evaluation criteria* need to be defined, which in our case are determined by the goals (activities as well as aspects). These criteria can range from simple activities to be supported to abstract aspects like usability. Evaluation criteria should on the one hand be understandable by the user; on the other hand be precise and comprehensive enough to capture the complexity of the tool system development and use. Second, the community doing the evaluation needs to be characterized: What are the actors and their goal concepts? Does the community evaluate as a whole, arriving at a single, majority or consensus evaluation result, or is it decomposed into groups or even individuals. How do their differences in evaluations play a role in the interpretation and decision making processes?

– Finally, *decision making* needs to take place based on the evaluation results. The decision making process needs to be examined from the point of view of stakeholders involved, the decision making procedure, and types of interventions in the socio-technical system. Who makes the decisions on the tool system design and implementation: the users and interpreters, or external authorities? Is the decision making procedure based on quantitative results only, or are also qualitative descriptions taken into account? Is there any interaction between interpreters and decision makers? How are the results used to decide on the selection/acquisition, configuration, or integration of new software?

5.2 The Scoring Process

The subprocesses sketched above can be implemented in many different ways. We do not aim to provide one particular approach here, since our goal is really to sketch the framework in which to position these processes. However, we will examine one crucial subprocess, the *scoring process*, in more depth here, as it is at the heart of the interpretation process. This, in turn, is the main process in which stakeholders reflect on the roles of the various technologies in their complex usage context.

In [10], we presented a scoring method based on Bedell's method for the evaluation of IT functionality effectiveness [4]. In Bedell's method, IS functionalities are scored on both the effectiveness and importance for the activities they support. The activities themselves are also scored on their importance to the organization. The importance scores act as weights for the effectiveness scores. Bedell has developed a whole range of increasingly aggregate indicators, the most general one being whether as an organization to invest in ICT at all. Through these indicators, the method is a useful aid in the organizational ICT interpretation and decision making process.

Simplifying Bedell's approach, we proposed a practical method for courseware evaluation in [10]. This approach is applicable to any tool system, however. Rather

than using Bedell's elaborate and quite complex framework of indicators, we limited ourselves to two indicators: *activity scores* and *functionality scores*. These indicators can be used to answer two basic questions: (1) how well are the various activities supported by the functionality components? (2) how effectively are the various functionality components used? In the current paper, we keep the notion of functionality scores, but generalize the activity scores to *goal scores*, as not only activities but also cross-activity aspects are possible sources of purpose to be taken into account in evaluation. We therefore consider both activity scores and *aspect scores* to be subtypes of goal scores.

Goal Scores

Goal (activity/aspect) scores show how useful the combined functionality components are for the support of a particular goal. Functionality scores represent the usefulness of a particular functionality component in supporting the combined goals of a community. In calculating these scores, the basic elements to be defined by the users in their actor roles are:

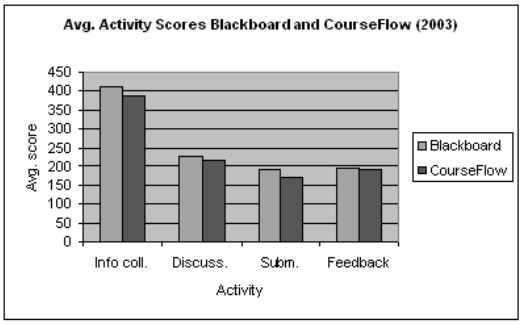

Fig. 4. Activity Scores for Blackboard and CourseFlow in 2003

- $I(g)$ = importance of a goal
- $I(f, g)$ = importance of a functionality in supporting a particular goal
- $Q(f, g)$ = quality of a functionality in supporting a particular goal.

Users can define their own criteria for explicating the importance and quality scores, or leave them implicit, just using their intuition. Whether they decide to further explicate their assessments or not, our approach allows (individual) evaluations to be positioned and aggregated in the larger evaluation framework and process.

(i) *G-Score* = $\sum I(f_i, g) * Q(f_i, g)$, for all functionalities 1..i

For all functionality components f_i supporting a particular goal g, the experienced quality of the support they provide, is multiplied by their importance in supporting this

activity. The sum of these values measures the usefulness of the combined tool components for a particular goal to the scoring user. This measure is especially useful for technology users, such as lecturers and students, or program chairs.

(ii) *F-Score* = $\sum I(g_j) * I(f, g_j) * Q(f, g_j)$, for all goals 1..j

For all goals g_j supported by a particular functionality component f, the quality of the support provided is multiplied by its importance for this activity *and* by the importance of the goal. This last multiplication is necessary as support for goals more important to the users should weigh more than for less relevant goals. The sum of these values measures the usefulness of a particular functionality component for the combined goals of the scoring user(s). This measure is especially useful for technology maintainers and developers. It allows, for example, a university computer centre or corporate IT department to determine which tool system components to get, or to get rid of.

An Example: Evaluating the Making of Group Assignments with Courseware

In 2002 and 2003, the approach was experimented with by Information Management students at Tilburg University. In both years, they tested the value of different Blackboard modules for the making of group assignments. This process was subdivided into four activities (information collection, group discussion, submission of results, and feedback from peers), while 11 functionality modules were examined (Send E-Mail, Discussion Board, Virtual Chat, etc.). In 2003, also an open source courseware tool, CourseFlow, was scored on similar modules. The results were interpreted by the software manager of Tilburg University, who considered them useful as an input in future decision making for courseware acquisition.

In this paper, we are not going in depth into the details of the scoring process. A detailed account of the goals, design, and results of the experiment is given in [10]. From that paper, we show two figures to give the reader a feel for how the process works. Fig. 4 shows how similar both tools are in their support for group assignment making. Fig. 5 shows how similar both tools in the experienced usefulness of their functionality components. This is powerful evidence for the usefulness of open source courseware, at least from the point of view of students. Given that much open source software is actually more reliable and secure than proprietary software, from the point of view of the computer center as well, open source courseware could be a viable alternative to expensive vendor software. By making careful evaluations of which activities need to be supported by which particular functionalities, the funds then becoming available might be put to much better use, for example to develop functionality for needs that currently are not supported. To give an indication, Tilburg University alone, with about 10,000 students, paid many tens of thousands of euros in license fees for the use of Blackboard courseware in 2003. Given the strong collaboration of Dutch universities in the inter-university ICT and networking organization SURFNet, the combined funds for a joint courseware product development and evaluation project might easily add up to many hundreds of thousands of euros annually. Many universities already have an installed base of (proprietary) courseware applications. It is not feasible to completely change

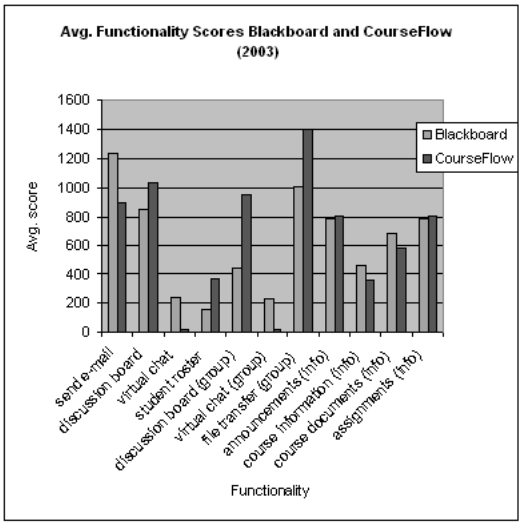

Fig. 5. Functionality Scores for Blackboard and CourseFlow in 2003

applications. However, through plug-ins and links, specialized functionality modules can be easily added to applications. Courseware evaluation of components should contribute to detect the gaps which such specialized modules could fill.

Our approach only provides a bird's eye view. Very high scores indicate high quality and high importance; very low scores indicate low quality and low importance. To interpret intermediate results correctly, the source data need to be analyzed in more detail, and to ensure the significance of many of the smaller differences, statistical tests such as t-test comparisons would be needed . However, given the complexity of the evaluation task, and the many different opinions of what are relevant evaluation criteria, false precision would not add much value. Instead, recognition of the broad patterns as a starting point for focused discussion and analysis is what a method like ours can contribute

The particular results may not be completely generalizable to other student populations, as Information Management students may have different preferences and evaluation criteria given their personal interest in advanced information technologies. Furthermore, courseware may also be used for many different purposes. In this paper, we analyzed only the group assignment handling process, but scores are probably very different when the method is applied to other workflows.

6 Discussion

Besides being limited in scope with respect to results, the scoring process proposed is still primitive in methodological functionality. Possible extensions include user-friendly versions of Bedell's [4] more advanced concepts, allowing for more refined judgments in the intermediate range of scores. Besides just giving a generic quality score, specific

quality aspects such as user friendliness, effectiveness, and maintainability could be included. Such indicators themselves also need to be interlinked, for example to ensure that different indicators measure the same quality aspect, thus increasing the value of measurement [35]. However, we should be careful not to introduce too much methodological complexity, as it scares away intended users and makes interpretation of evaluation results by the users themselves difficult. Careful consideration of the trade-off between methodological power and comprehensibility, for example by experimentation with different types of scores and visualizations, might considerably increase the value of a practical evaluation method (e.g. [16]).

Another limitation of the example approach presented is that only students were asked to evaluate courseware functionality. Still, their aggregated evaluations were used in decision making by other stakeholders, such as the software manager. A more comprehensive approach thus would have to contrast evaluations by different stakeholders, since the interests of all stakeholders need to be balanced. For example, what is the software manager to do when a lecturer evaluates a functionality shared with students very differently? Whose evaluation to give preference? To address these important issues, techniques to identify and manage stakeholders interests, and to design high-quality multi-stakeholder dialogues aimed at improved relationship building and learning potentials could play a key role [2, 23].

The IS quality literature has produced many evaluation criteria frameworks, covering more or less completely the information systems development process. One classic example is Delen and Rijsenbrij's quality tree, which decomposes information systems quality into 46 quality attributes, organized in four dimensions: the process dimension, concerning the development process; the static dimension, which refers to the control of the IS; the dynamic dimension, covering the functioning of the system as perceived by the user; and the information dimension, concerning the quality of the information itself [11]. Along similar lines, DeLone and McLean, in their famous survey article of the literature, identified a very large, confusing and partially overlapping set of IS "success" measures [12]. Whereas business process models and workflow patterns can extend the ontologies of activity-type goals in our evaluation approach, such quality frameworks can do the same for the aspect-type goals.

The weighing, comparison, and aggregation of scores is more complex than sketched here: for example, there are many dependencies between goals. Also, can activities and aspects be compared in one measure, or should they be measured separately? To refine such aspects, pragmatic evaluation could also benefit from the quality IS literature. Many evaluation methods have been designed, so many that their classification has become a research topic in its own. For example, based on whether an evaluation method has a financial or non-financial focus and the type of parameter calculation technique used, Renkema and Berghout distinguish between financial, multi-criteria, ratio, and portfolio approaches [26]. Typical examples of financial methods are calculations of Net Present Value and Internal Rate of Return. However, since just focusing on financial aspects is insufficient when looking at information systems with strategic impact, some explicit process needs to be designed to make intangibles measurable as well [19]. Multi-criteria approaches allow for non-financial aspects to be quantified, preferably leading to aggregate measures for each investment. For example, in their In-

formation Economics method, Parker et al. distinguish between different non-financial aspects from the business and technology domains, such as 'strategic match' and 'IS infrastructure risk' [22]. These aspects get weighed and scored, leading to subtotals of positive and negative weighted scores, which give an indication of the overall value of the proposal. Related methods like Quality Function Deployment (QFD) help to translate customer needs into balanced sets of relevant design and production requirements [6].

A different classification of evaluation methods is proposed by Bannister and Remenyi [3]. They distinguish between fundamental measures, metrics which parameterize characteristics down to a single measure, and composite approaches (such as Information Economics and Bedell's method), which combine several fundamental measures to obtain an overall picture of value/investment return. However, they go beyond the calculation of the metrics and examine their interpretation. To this purpose, they introduce a third category of approaches, meta approaches, which attempt to select the optimum set of measures for a context or set of circumstances. Furthermore, they note that fundamental, composite, and meta-approaches can be applied in two different ways. Many methodologies take a positive/reductionist stance, where the decision maker accepts the best overall score without questioning. Hermeneutic perspectives, on the other hand, allow for the decision maker to interpret several metrics in a way that cannot be formally stated. According to the authors, these approaches are the more interesting, for example by letting "instinct" play a more important role [3]. In our case, it would be particularly interesting to examine work on hermeneutic meta-approaches, to develop the kind of pragmatic evaluation methods needed for interoperable tool systems. Others propose that general evaluation methods focusing on multi-criteria comparison like QFD should be embedded in a group decision support process to become really useful [6]. Thus, finding the right balanced between formal and informal evaluation, in particular the link between informal interpretation and decision making with quantitative scoring approaches needs serious investigation. Conceptual structures tools could play a very important role here, by allowing for evaluation process ontologies to be defined and used, defining patterns at the right level of granularity between total formality and complete informality.

The use of patterns is a very important extension of the evaluation approach proposed. In [8], we examined the roles of individual and communal conceptual patterns in designing the Pragmatic Web. We proposed a typology of collaboration patterns in [9]. Such patterns can enrich the modeling of the usage context and tool system and the mappings between and within these systems. Conceptual structures tools can be very helpful, not as operational tools to support collaboration processes directly, but as *meta*-tools for defining collaborative requirements and the tool functionalities that can satisfy these requirements. They can support the capture, representation, and analysis of the collaboration patterns that are the essence of the evolving socio-technical system formed by a collaborative community and its supporting tool system. One useful extension could be to refine our current simple activity-concept with more complex and realistic business process and workflow ideas. Vice versa, business process/workflow management systems could benefit from more systematic evaluation. For example, much interesting work is happening on such standards like the Business Process Management

Notation (BPMN) for modelling business processes and, for instance, the Business Process Execution Language for Web Services (BPEL4WS) for describing the design of and the mappings to the systems supporting such processes [36]. Which services to use for what business processes is unclear, however. Our approach could help conceptualize evaluation processes. A related notion are workflow patterns, which can be seen from many perspectives, e.g. the control flow, data, resource, and exception handling perspectives[31]. Many control flow structures, for example, have to do with temporal or role restrictions on flow elements[6]. It would be worth examining to what extent such a canon of workflow patterns could be mapped to particular tool system components, and at what level of granularity.

In this paper, we examined the notion of 'pragmatics' from a very basic goal-orientation perspective. In philosophy, much more advanced treatments of this concept have been proposed, many of them too theoretical to be useful for the messy practice of systems development. In particular in our Conceptual Structures research community, however, interesting attempts at applying pragmatics-philosophy to the construction of concrete tool systems development methodologies are underway. Examples are Keeler and Pfeiffer's interest in a Peirce-grounded pragmatic methodology for the development of scientific collaboratories and knowledge representation systems [18], Richmond's trikonic architectonic for inter-enterprise systems evolution [27], Delugach's active knowledge systems architecture [13], and the goal-oriented transaction modelling in multi-agent systems proposed by Polovina et al [25]. Related work from the field of argumentation theory shows how pragmatics can inform design processes of socio-technical systems [1].

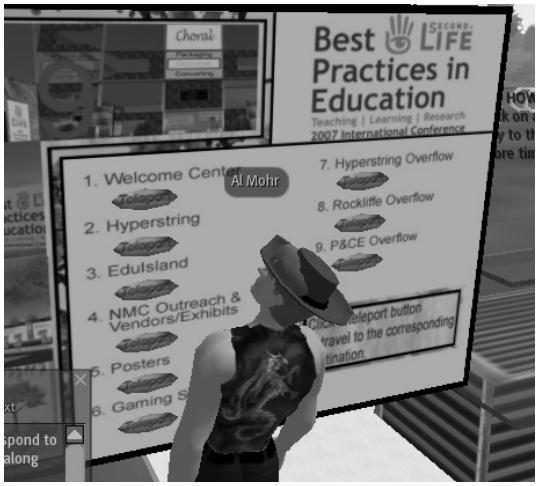

Fig. 6. The Author/Al Mohr Attending the First Second Life International Conference

[6] See http://www.workflowpatterns.com/patterns/ for many examples of these and other workflow patterns

So far, we have been looking at tool interoperability in the "real" world. However, a whole new class of tool systems is rapidly emerging in *virtual worlds* like Second Life[7]. Until recently, they were still in their infancy due to hardware and software limitations. However, as these worlds are rapidly maturing, an increasing number of serious applications are developing. An exciting recent event was the First Second Life Best Practices in Education International Conference[8]. This conference was held entirely "in-world", allowing people from all over the world to attend conference and poster sessions, vendor exhibits and so on using an avatar, like this picture of the author/Al Mohr shows (Fig. 6). These worlds are also very significant from a tool system interoperability point of view as they are filled with actionable objects. In fact, everything in such a world is a virtual object, which can show behavior through scripts. These objects themselves can and are being combined into ever more complex systems, which display very sophisticated overall behaviors, many of which are impossible in the Real World. For example, one of the favorite modes of movement in Second Life is flying. Not using a plane, but by persons just floating away from the ground without external help. To develop effective collaborative systems in-world, pragmatic evaluation of such "virtual object systems" could lead to as of yet unimaginable new objects and behaviors, even the sky no longer being the limit.

7 Conclusions

Choosing the right functionality components to include in a tool system is not simply a technological decision, but a strategic socio-technical development choice. Functionality selection requires a careful balancing of the multiple professional, political, and social requirements of a collaborative community with the affordances and constraints inherent in the interoperable tool system. Without adequate, ongoing tool system evaluation, the socio-technical gap between work practices and supporting tools may easily become too large, endangering collaboration.

In this paper, we examined the pragmatic evaluation of tool system interoperability. We have explored the make-up and links between the tool system and usage context, and examined the process in which the socio-technical calibration between these two subsystems needs to take place. We did not present an operational evaluation method, but outlined a conceptual framework for characterizing and comparing pragmatic evaluation methods. Instead, our aim was to convey the complexity of the kind of socio-technical systems analysis needed.

It is not sufficient to merely produce tools that *could* interoperate at the technical or semantic level. Instead, it is essential that tool components are evaluated in many different community contexts-of-use, in order to find out what interoperations of tool components *do* work in practice. Two major problems with such pragmatic evaluation, however, are the infinite variety of usage contexts, and the subtle balance that needs to be found between formal representations and human, informal, interpretation. As Pacey puts it: "tacit knowledge in a human mind not only is operational knowledge that can

[7] http://www.secondlife.com

[8] http://slbestpractices2007.wikispaces.com/. A photo gallery of one of the conference sessions can be found at http://communitysense.net/info/?q=node/53

easily be built into a machine, but also includes a sense of what the knowledge means and how it is related to human purposes [21, p.9]". Conceptual structures tools could be the key to linking the unique human capacity to interpret the meaning of an infinity of contexts with computational power. These tools having the ability to represent and reason about patterns, combined with their strong focus on syntactic *and* semantic interoperability, make them an indispensable component in the next generation pragmatic evaluation methods. They will pave the way to a gradual meshing of method and system, as the information systems of the future will be continuously evolving, driven by the purposeful meta-systems formed by their communities of use in combination with the analytical power of conceptual structures tools.

References

1. M. Aakhus and S. Jackson. Technology, interaction, and design. In K.L. Fitch and R.E. Sanders, editors, *Handbook of Language and Social Interaction*, pages 411–435. Lawrence Erlbaum, Mahwah, NJ, 2005.
2. F. Ackermann and C. Eden. Powerful and interested stakeholders matter: Their identification and management. In *Proc. of the Association of Management Conference, Seattle, August 4-6, 2003*, 2003.
3. F. Bannister and D. Remenyi. Acts of faith: Instinct, value, and IT investment decisions. *Journal of Information Technology*, 15(3):231–241, 2000.
4. E. Bedell. *The Computer Solution: Strategies for Success in the Information Age*. Dow Jones-Irwin, Homewood, Ill., 1985.
5. E. Berghout. A dilemma between decision quality and confidence in the decision: Experimental validation of investment analysis methods. *Electronic Journal of Information Systems Evaluation*, 5(1), 2001.
6. G. Büyüközkan and O. Feyzioglu. Group decision making to better respond customer needs in software development. *Computers & Industrial Engineering*, 48(2):427–441, 2005.
7. S. Christiaens and A. de Moor. Tool interoperability from the trenches: the case of DOGMA-MESS. In A. de Moor, S. Polovina, and H. Delugach, editors, *Proc. of the First Conceptual Structures Tool Interoperability Workshop*, pages 103–118. Aalborg University Press, 2006.
8. A. de Moor. Patterns for the Pragmatic Web. In *Proc. of the 13th International Conference on Conceptual Structures (ICCS 2005), Kassel, Germany, July 2005*, pages 1–18, 2005.
9. A. de Moor. Community memory activation with collaboration patterns. In *Proc. of the 3rd Prato International Community Informatics Conference (CIRN 2006), Prato, Italy, 9-11 October, 2006*, 2006.
10. A. de Moor. A practical method for courseware evaluation. In *the Second International Conference on the Pragmatic Web (PragWeb 2007), Tilburg, The Netherlands, 22-23 October 2007, submitted*, 2007.
11. G.P.A.J. Delen and D.B.B. Rijsenbrij. The specification, engineering, and measurement of information systems quality. *Journal of Systems and Software*, 17:205–217, 1992.
12. W.H. DeLone and E.R. McLean. Information systems success: The quest for the dependent variable. *Information Systems Research*, 3(1):60–95, 1992.
13. H. Delugach. Towards building active knowledge systems with conceptual graphs. In *Proc. of the 11th International Conference on Conceptual Structures (ICCS 2003), Dresden, July 2003*, pages 296–308, 2003.
14. P.N. Edwards, S.J. Jackson, G.C. Bowker, and C.P. Knobel. Understanding infrastructure: Dynamics, tensions, and design - report of a workshop on "History & theory of infrastructure: Lessons for new scientific cyberinfrastructures". Technical report, 2007.

15. D. Engelbart. Coordinated information services for a discipline- or mission-oriented community. In *Proc. of the 2nd Annual Computer Communications Conference, San Jose, California, January 24*, 1973.
16. T. Erickson, C. Halverson, W.A. Kellogg, M. Laff, and T. Wolf. Social translucence: Designing social infrastructures that make collective activity visible. *Communications of the ACM*, 45(4):40–44, 2002.
17. P. Fremantle, S. Weerawarana, and R. Khalaf. Enterprise services. *Communications of the ACM*, 45(10):77–82, 2002.
18. M.A. Keeler and H.D. Pfeiffer. Building a pragmatic methodology for KR tool research and development. In *Proc. of the 14th International Conference on Conceptual Structures (ICCS 2006), Aalborg, Denmark, July 16-21, 2006*, pages 314–330, 2006.
19. K.E. Murphy and S.J. Simon. Intangible benefits valuation in ERP projects. *Information Systems Journal*, 12:301–320, 2002.
20. I. Nonaka, R. Toyama, and N. Konno. SECI, ba and leadership: A unified model of dynamic knowledge creation. *Long Range Planning*, 33:5–34, 2000.
21. A. Pacey. *Meaning in Technology*. The MIT Press, Cambridge, MA, 2001.
22. M.M. Parker, R.S. Benson, and H.E. Trainor. *Information Economics*. Prentice Hall, 1988.
23. S.L. Payne and J.M. Calton. Exploring research potentials and applications for multi-stakeholder learning dialogues. *Journal of Business Ethics*, pages 71–78, 2004.
24. A.V. Pietarinen. Peirce's theory of communication and its contemporary relevance. In K. Nyiri, editor, *Mobile Learning: Essays of Philosophy, Psychology and Education*, pages 46–66. Passagen Verlag, Vienna, 2003.
25. S. Polovina and R. Hill. Enhancing the initial requirements capture of multi-agent systems through conceptual graphs. In *Proc. of the 13th International Conference on Conceptual Structures (ICCS 2005), Kassel, Germany, July, 2005*, pages 439–452, 2005.
26. T. Renkema and E. Berghout. Methodologies for information system investment evaluation at the proposal stage: A comparative review. *Information and Software Technology*, 39(1):1–13, 1997.
27. G. Richmond. Trikonic architectonic. In *Proc. of the 15th International Conference on Conceptual Structures (ICCS 2007), Sheffield, UK, July 22-27, 2007*, 2007.
28. S. Sawyer. A market-based perspective on information systems development. *Communications of the ACM*, 44(11):97–102, 2001.
29. A.P. Sheth. Changing focus on interoperability in information systems: From system, syntax, structure to semantics. In M.F. Goodchild, M.J. Egenhofer, R. Fegeas, and C.A. Kottman, editors, *Interoperating Geographic Information Systems*, pages 5–30. Kluwer, 1998.
30. M.P. Singh. The Pragmatic Web. *IEEE Internet Computing*, 6(3):4–5, 2002.
31. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14:5–51, July 2003.
32. P. van Schaik. Involving users in the specification of functionality using scenarios and model-based evaluation. *Behaviour & Information Technology*, 18(6):455–466, 1999.
33. G. Vetere and M. Lenzerini. Models for semantic interoperability in service-oriented architectures. *IBM Systems Journal*, 44(4):887–903, 2005.
34. A.E. Walsh. *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book*. Prentice Hall, 2002.
35. R. Watts. *Manufacturing Software Quality*. NCC Publications, Manchester, 1987.
36. S.A. White. Introduction to BPMN. Technical report, 2004.
37. T. Winograd and F. Flores. *Understanding Computers and Cognition - A New Foundation for Design*. Ablex Publishing Corporation, 1986.